IN THE CLAIMS:

The following listing of claims will replace all prior versions, and listings, of claims in the application.

1-43. (Cancelled)

44. (New) A method for write-back of modified graphics data, the method comprising:
    a) ordering a list of data blocks currently in a level one cache by a least recently used value determined for each data block;
    b) setting a pointer to point to a data block at the top of the list, wherein the data block at the top of the list has a largest least recently used value, and wherein a data block currently indicated by the pointer is a current data block;
    c) testing a dirty tag bit corresponding to the current data block, wherein the dirty tag bit indicates whether the data in the block is modified;
    d) setting the pointer to point to a next data block in the list, if the dirty tag bit indicates that the current data block is not modified, and returning to c);
    e) issuing a command to a memory request processor to write-back the current data block from the level one cache to a corresponding level two cache data block, if the dirty tag bit indicates that the current data block is modified;
    f) modifying the dirty tag bit corresponding to the current data block to indicate that the current data block is no longer modified and the memory locations are available for future allocation; and
    g) setting the pointer to point to a next data block in the list, and repeating steps c) through f) for each of the remaining data blocks in the list.

45. (New) The method of claim 44, further comprising determining a least recently used value for each data block in the level one cache prior to a).

46. (New) The method of claim 44, further comprising re-determining a least recently used value for each data block in the level one cache after the data block has been accessed.

47. (New) The method of claim 44, further comprising stalling e) until an empty memory cycle is detected.

48. (New) The method of claim 47, wherein said empty memory cycle is detected on each buss connected to the level one cache and also on each buss connected to a level two cache with a data block corresponding to the current data block.

49. (New) The method of claim 47, wherein said stalling is ignored in response to an urgent request for allocation of memory within the temporary storage space.

50. (New) The method of claim 44, wherein the level two cache is configured as a write-through cache, and as the current data block is written to the level two cache, it is also written to an associated DRAM memory bank connected to the level two cache.

51. (New) The method of claim 44, wherein the modified graphics data is sample data, and wherein a set of sample data is filtered to determine color values for a corresponding pixel.

52. (New) The method of claim 44, wherein the set of sample data includes sample data for samples located within a filter region corresponding to a location of the corresponding pixel.

53. (New) The method of claim 44, wherein the least recently active value determined for each data block is based on how recently the data block was accessed.

54. (New) The method of claim 44, wherein an arithmetic logic unit connected to the level one cache memory is configured for:

receiving as a first operand graphics data from a source external to the memory;

receiving as a second operand graphics data stored in the level one cache;

combining arithmetically the two operands according to a function defined by an external control signal; and

storing the resulting modified graphics data in the level one cache.

55. (New) The method of claim 44, wherein a) through g) are repeated periodically.

56. (New) The method of claim 44, wherein a) through g) are repeated continuously.

57. (New) A graphics system comprising:

one or more memories configured to receive and store graphics data, wherein each memory comprises, on a single integrated chip:

one or more RAM memories configured to store the graphics data;

a level two cache connected to each RAM memory; and

a level one cache connected to each of the level two cache memories;

an array of registers for each of the one or more memories configured to store status information, wherein the status information indicates, for each block of memory in the corresponding level one cache, whether the graphics data is modified or unmodified and a least recently used value; and

a memory interface connected to the one or more memories and to the array of registers, wherein the memory interface is operable to:

a) store a list of data blocks currently in the level one cache;

b) order the list by the least recently used values for each data block;

c) set a pointer to point to a data block at the top of the list, said pointer thereby identifying a current data block;

d) copy graphics data from the current data block to a corresponding level two cache data block if the status information for the current data block indicates the graphics data is modified;

e) reset the pointer to point to a next data block in the list, if the status information indicates the current data block is not modified, and return to d);

f) modify the status information corresponding to the current data block to indicate that the current data block is no longer modified and the memory locations are available for future allocation; and

g) reset the pointer to point to a next data block in the list, and repeat d) through g) for each of the remaining data blocks in the list.

58. (New) The graphics system of claim 57, further comprising a set of busses connected to the level one and level two caches, wherein the memory interface is further operable to stall said copy graphics data until an empty memory cycle is detected on each of the busses connected to the level one cache and the level two cache with the corresponding data block.

59. (New) The graphics system of claim 57, wherein the data block at the top of the list has a largest least recently used value.

60. (New) The graphics system of claim 57, wherein the array of registers stores status information indicative of the current state of each level one cache plus the predicted results of one or more memory requests pending in the request queue, and wherein the least recently used values reflect the pending memory requests.

61. (New) The graphics system of claim 57, wherein each memory further comprises a shift register connected to each RAM, wherein each shift register is configured to receive and store portions of the graphics data from each RAM, and wherein each shift register is further configured to output graphics data serially in response to an external clock signal.

62. (New) The graphics system of claim 57, further comprising a video processor and a display device, wherein the display device displays images according to the video data generated by video processor from the graphics data.

63. (New) The graphics system of claim 57, wherein each memory further comprises an arithmetic logic unit connected to the level one cache memory and configured to:
receive as a first operand graphics data from a source external to the memory;
receive as a second operand graphics data stored in the level one cache;
arithmetically combine the two operands according to a function defined by an external control signal; and
store the results of the arithmetic combination in the level one cache.